3. RFID Tutorial: Introduction to RFID data filtering

Gabrielle Davidson - School of Biological Sciences, University of East Anglia

2024-08-12

3. Get set up

Open R Studio

Check R packages installed

```
#load the following packages
library(tidyverse)
library(rmarkdown)
library(dplyr)
library(tidyr)
library(lubridate)
library(ggplot2)
library(stringi) #may not need this
library(magrittr) #may not need this
```

Download resources necessary for the workshop

Download the resources from **my github page (https://github.com/DrGLDavidson/RFID-workshop)**

3.1 Import your dataset to your environment from your working directory.

```
require("knitr")
opts_knit$set(root.dir = "F:/RWorkspace/GitHub/RFID-workshop/data/outputFiles")

#clear the global environment so we don't have any conflicts with the next steps

rm(list = ls(all.names = TRUE))

#choose the appropriate working directory

setwd("F:/RWorkspace/GitHub/RFID-workshop/data/outputFiles")

#call your most recent dataset of merged feeder data.

df<-read.delim("Masterdf_220209.txt", header=TRUE)
head(df)
```

```
##    F               Date Hmsec     ID Event Channel    Dur  Clks   Freq Edges
## 1 D  2022-02-04 09:16:45   196 c1935     82       0 114855 61559 126720     1
## 2 D  2022-02-04 09:17:11   896 c1935     83       0 106855     0 125184     0
## 3 D  2022-02-04 09:17:11   648 c1935     83       0 114855    27 126208   156
## 4 D  2022-02-04 09:17:12   650 c1935     83       0 106855    37 126208   843
## 5 D  2022-02-04 09:18:30   270 c1935     84       0      2 62210 126208     1
## 6 D  2022-02-04 09:20:09   667 c1935     85       0 119946 17899 126464     1
##   Reps Type  TagID_hex Mfr TagID feeder
## 1    1    0             NA    NA    F01
## 2    0    0             NA    NA    F01
## 3    1    1 0300024FEF  NA    NA    F01
## 4    1    0             NA    NA    F01
## 5    1    0             NA    NA    F01
## 6    1    0             NA    NA    F01
```

```
names(df)
```

```
##  [1] "F"         "Date"      "Hmsec"     "ID"        "Event"     "Channel"  
##  [7] "Dur"       "Clks"      "Freq"      "Edges"     "Reps"      "Type"     
## [13] "TagID_hex" "Mfr"       "TagID"     "feeder"
```

3.2 Data filtering

3.2.1 Remove misreads (i.e. erroneous TagID_hex values)

Ocassionally the RFID reader will incorrectly read an tag. This may be because the bird does not properly land on the perch to feed, and so these 'misreads' can be removed. To remove misreads, we need to match the feeder data against known tagged bird IDs (i.e. Passive Integrative Transponder (PIT) tag numbers, aka TagID_hex). I have provided a file with a list of PIT tags and corresponding metadata (species, age, sex etc.). Note this data is NOT from the UEA population.

```
#upload a list of your known PIT tag IDs. Notice that the file is a .csv, rather than .txt
PIT<-read.csv(file="F:/RWorkspace/GitHub/RFID-workshop/data/PITList_tutorial.csv", header=TRUE)
names(PIT)
```

```
##  [1] "date"        "time"        "btoRingType" "btoID"       "species"    
##  [6] "pitTYPE"     "pitID"       "age"         "sex"         "wing"       
## [11] "weight"
```

```
#note that the ring type (BTO and PIT) refer to whether it was a new bird (first time given the
ring), or retrap (already had the ring). It is possible that a bird may already have a BTO ring
(R) but not a PIT tag (N). However, in these cases, the code was meant to be A for added (but y
ou can ignore this detail for now).

#check how many unique PIT tag IDs there are, it should match the number of observations (globa
l environment). If it does not match, it suggests duplicates. Duplicates would matter if you ar
e planning to merge the metadata and you want it to be a particular date (e.g. age of a bird wh
en it was first fitted with a ring)

length(unique(PIT$pitID))
```

```
## [1] 4033
```

```
#to merge/match PIT tags from two files you must ensure the header that contains the values of interest are named the same across files. Currently they are not.

head(PIT)
```

```
##         date  time btoRingType   btoID species pitTYPE      pitID age sex wing
## 1 02/01/2023 11:17           R AJT8236      BT       A 3B0018373B   5       65
## 2 13/10/2022 12:46           C AYN4586      BT       A 3B00184BED   3       61
## 3 29/09/2022 09:33           R TV35268      GT       A 3B00187309   3   M   77
## 4 09/11/2021 09:59           R AAJ5735      BT       A 3B0018FA87   3       62
## 5 09/11/2021 11:54           R AAJ5802      BT       A 3B00193EAD   3       63
## 6 07/11/2021 11:26           R AJT8125      BT       A 3B001904FD   3       66
##   weight
## 1   11.5
## 2    9.8
## 3   19.3
## 4   10.9
## 5   10.9
## 6   10.8
```

```
head(df)
```

```
##    F                Date Hmsec     ID Event Channel    Dur  Clks   Freq Edges
## 1 D  2022-02-04 09:16:45   196 c1935     82       0 114855 61559 126720     1
## 2 D  2022-02-04 09:17:11   896 c1935     83       0 106855     0 125184     0
## 3 D  2022-02-04 09:17:11   648 c1935     83       0 114855    27 126208   156
## 4 D  2022-02-04 09:17:12   650 c1935     83       0 106855    37 126208   843
## 5 D  2022-02-04 09:18:30   270 c1935     84       0      2 62210 126208     1
## 6 D  2022-02-04 09:20:09   667 c1935     85       0 119946 17899 126464     1
##   Reps Type  TagID_hex Mfr TagID feeder
## 1    1    0            NA    NA    F01
## 2    0    0            NA    NA    F01
## 3    1    1 0300024FEF  NA    NA    F01
## 4    1    0            NA    NA    F01
## 5    1    0            NA    NA    F01
## 6    1    0            NA    NA    F01
```

```
#rename the header in PIT to match that of df
names(PIT)[names(PIT) == "pitID"] <- "TagID_hex"
names(PIT)
```

```
##  [1] "date"        "time"        "btoRingType" "btoID"       "species"
##  [6] "pitTYPE"     "TagID_hex"   "age"         "sex"         "wing"
## [11] "weight"
```

If we view df, we will also see there are a lot of rows with no TagID_hex. This is because non-tagged birds often land on the feeder and it shows up as a time stamp with no corresponding TagID_hex. Lets remove blank rows, otherwise there will be many rows that can’t merge with PIT.

```
df <- with(df, df[!(TagID_hex == "" | is.na(TagID_hex)), ])
```

Now you can merge dataframes by their shared TagID_hex value. In addition, if there are any missing TagID_hex values from PIT, it will be indicated with “NA”. In other words, if a bird that is not on our database was detected at the feeder, it will show up as NA. This code also returns any values from our PIT dataframe that was not on df. In otherwords, an NA is given if the bird on our database was never detected at the feeders.

```
df1 <- merge(df, PIT, by = "TagID_hex", all = TRUE)
df1[is.na(df1)] <- "NA"
```

#We are only interested in the TagID_hex that was detected at the feeder, but not on our database. We will filter these into a new dataframe called missing. We extract these as values that did not have a BTO

```
missing<-df1%>%
  filter(btoID=="NA")
```

why do we do this with the header btoRing? Because TagID_hex is now merged across dataframes, so this is no longer informative for missing data. Any of the headers from the PIT dataframe would work, but we are using btoID.

```
#there are 54 observations where there is no corresponding PIT tag.
#create a list of TagID_hex reads that did not have a match:

uniqueMisreads <- unique(missing$TagID_hex)
uniqueMisreads
```

```
## [1] "01103F3B1B" "0300024FEF"
```

there are two tags. “0300024FEF” is our reference tag. It is the tag that the experimenter uses to test the feeders and mark when experiments start and end. “01103F3B1B” looks like a genuine tag. But is it a misread or missing data from our PIT tag data? To answer this, double check the master ringing database to see if that tag is there. If it is not there, assume it is a misread. If it is there, add it to the PIT database and re-run your code.

```
#save your misreads for your records

path_out = 'F:/RWorkspace/GitHub/RFID-workshop/data/outputFiles'

write.csv(uniqueMisreads, file.path(path_out, 'uniqueMisreadsFeeders.csv'))

#assuming we have resolved this, merge df and PIT again, using a different code that wont keep
all NAs from the PIT file, but will remove NAs from the btoRing column, using the argument all
= FALSE

df1 <- merge(df, PIT, by = "TagID_hex", all = FALSE)

#this returns a dataframe that is 1214 observations.
1268-1214  #maths to check number of observations makes sense
```

```
## [1] 54
```

The above subtraction equals 54, which is the same number of observations from our dataframe “missing”. This makes sense.

Another way you can merge dataframes is using the package dplyr and the following code:

```
df2<-left_join(df, PIT ,by ="TagID_hex")

# and then we have to remove NAs with another chunk of code

df2<-df2[!is.na(df2$btoRing),]

write.table(df2, file = "Masterdf_noMisreads.txt",sep="\t",row.names=FALSE)

#clear the global environment so we don't have any conflicts with the next steps

rm(list = ls(all.names = TRUE))
```

3.2.1 Filter repetitive RFID reads within a single visit

A common goal in quantifying bird behaviour is to extract the number of visits a bird makes to a feeder. Due to the nature of the RFID device, multiple rows will be recorded if the bird sits on the perch while feeding, yet this should be considered a single visit.

```
#As we will be working with time, we need to check how our dataframe is named and organised
df2<-read.delim("Masterdf_noMisreads.txt", header=TRUE)

head(df2)
```

```
##    F                Date Hmsec     ID Event Channel    Dur Clks   Freq Edges
## 1 D  2022-02-04 10:01:13   467 c1935    146       0      0   32 126464   264
## 2 D  2022-02-04 11:06:32   770 c1935    147       0      6   39 126208   426
## 3 D  2022-02-04 11:17:36   871 c1935    148       0 105573   51 126464  1168
## 4 D  2022-02-04 11:17:40   123 c1935    148       0 121015   27 126208   174
## 5 D  2022-02-04 11:17:41   946 c1935    148       0 121015   27 126208   566
## 6 D  2022-02-04 11:17:43   794 c1935    148       0 119945   35 125952   272
##   Reps Type  TagID_hex Mfr TagID feeder       date  time btoRingType   btoID
## 1    2    1 01103FC949  NA    NA    F01 09/01/2021 11:03           R AKL0680
## 2    3    1 01103F7DB1  NA    NA    F01 03/10/2021 10:30           R AAJ5894
## 3    5    1 01103F3BED  NA    NA    F01 17/10/2021 10:09           R AAJ5895
## 4    1    1 01103F3BED  NA    NA    F01 17/10/2021 10:09           R AAJ5895
## 5    1    1 01103F3BED  NA    NA    F01 17/10/2021 10:09           R AAJ5895
## 6    2    1 01103F3BED  NA    NA    F01 17/10/2021 10:09           R AAJ5895
##   species pitTYPE age sex wing weight
## 1      BT       R   5   M   64   11.7
## 2      GT       R   3   F   73   16.3
## 3      GT       R   3   M   75   18.5
## 4      GT       R   3   M   75   18.5
## 5      GT       R   3   M   75   18.5
## 6      GT       R   3   M   75   18.5
```

There are two separate date columns and a time column. This is because we merged the feeder data with ringing data.

```
#rename ringing columns

names(df2)[names(df2) == "date"] <- "dateRinged"
names(df2)[names(df2) == "time"] <- "timeRinged"

#rename feeder date column to indicate it also includes hours, minutes and seconds

names(df2)[names(df2) == "Date"] <- "dateTime"

names(df2)
```

```
##  [1] "F"           "dateTime"    "Hmsec"       "ID"          "Event"      
##  [6] "Channel"     "Dur"         "Clks"        "Freq"        "Edges"      
## [11] "Reps"        "Type"        "TagID_hex"   "Mfr"         "TagID"      
## [16] "feeder"      "dateRinged"  "timeRinged"  "btoRingType" "btoID"      
## [21] "species"     "pitTYPE"     "age"         "sex"         "wing"       
## [26] "weight"
```

create a new column with the date only

```
df2$date<-date(df2$dateTime)

#cleanup the dataframe by selected the columns and order we want them to appear

df2<-df2%>%
  select(date, dateTime, Hmsec, ID, Event, Channel, Dur, Clks, Freq, Edges, Reps, Type, TagID_h
ex, feeder, dateRinged, timeRinged, btoRingType, btoID, species, pitTYPE, age, sex, wing, weigh
t)
names(df2)
```

```
##  [1] "date"        "dateTime"    "Hmsec"       "ID"          "Event"      
##  [6] "Channel"     "Dur"         "Clks"        "Freq"        "Edges"      
## [11] "Reps"        "Type"        "TagID_hex"   "feeder"      "dateRinged" 
## [16] "timeRinged"  "btoRingType" "btoID"       "species"     "pitTYPE"    
## [21] "age"         "sex"         "wing"        "weight"
```

```
#we need to change the class of the timeDate column as a POSIX class.

df3<-df2%>%
  mutate(dateTime=ymd_hms(dateTime))

class(df3$dateTime)
```

```
## [1] "POSIXct" "POSIXt"
```

create a column that calculates the time difference with previous row of a dataframe grouped by date, feeder and RFID

consider why you are grouping by these columns. The time interval will only be calculated PER date PER feeder and PER tag. If you wanted to know the time interval between days as well, then you would not group by date. Or if you wanted an interval regardless or feeder, you would remove feeder from your grouping argument.

```
df4<-df3 %>%
  arrange(dateTime)%>%
  group_by(date, feeder, TagID_hex) %>%
  mutate(timeSincePreviousVisit = dateTime - lag(dateTime))%>%
  arrange(TagID_hex, feeder, dateTime)%>%
  ungroup()%>%
  select(feeder, TagID_hex,date, dateTime, timeSincePreviousVisit)

#look at the output of just those selected columns to see we have produced what we think we wan
t
View(df4)
#0 sec was calculated if the bird was on the feeder within the same seconds
#NA is given if it is the birds' first visit to feeder X for that date

#rerun the above code without the select() argument so we have the full dataframe of variables


df4<-df3 %>%
  arrange(dateTime)%>%
  group_by(date, feeder, TagID_hex) %>%
  mutate(timeSincePreviousVisit = dateTime - lag(dateTime))%>%
  arrange(TagID_hex, feeder, dateTime)%>%
  ungroup()



#the new column timeSincePreviousVisit has "secs" and we don't want that, so lets remove it

df4$timeSincePreviousVisit <- gsub(' secs', '', df4$timeSincePreviousVisit)

#the column timeSincePreviousVisit needs to be numeric for graphical purposes and for filtering
based on greater than/less than values

df4$timeSincePreviousVisit <- as.numeric(as.character(df4$timeSincePreviousVisit))
class(df4$timeSincePreviousVisit)
```

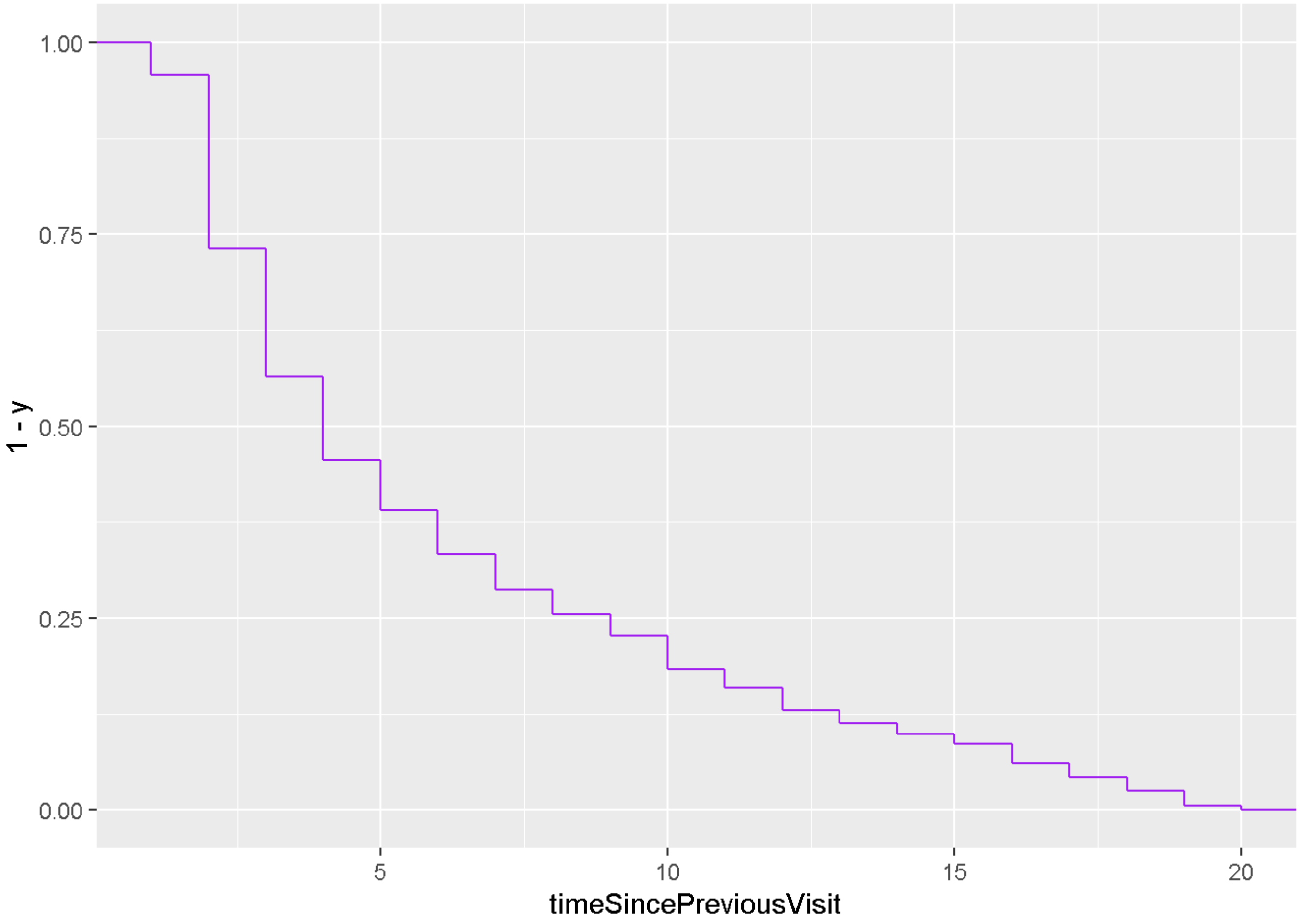
```
## [1] "numeric"
```

Create a dataframe of successive visits that are less than 20 seconds so we can graph how frequently birds are read at the feeder use the “select” argument to extract only the column named timesSincePreviousVisit

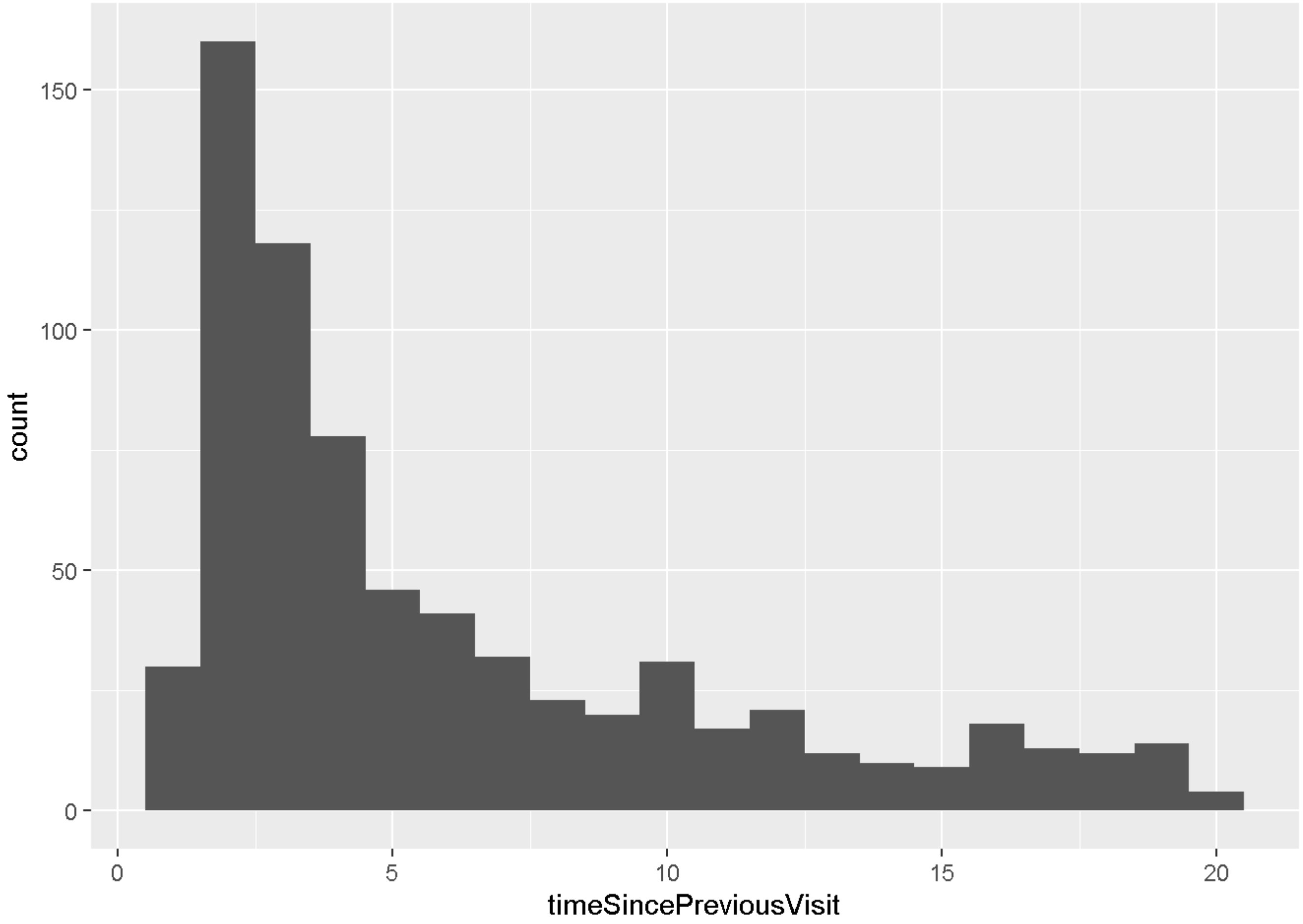
```
df5<-df4%>%
  filter(timeSincePreviousVisit <=20)%>%
  select(timeSincePreviousVisit)
class(df5$timeSincePreviousVisit)
```

```
## [1] "numeric"
```

```
#cumulative frequency graph
ggplot(df5, aes(timeSincePreviousVisit, y= 1-..y..))+
  stat_ecdf(geom = "step", color="purple")
```



```
#histogram
ggplot(df5, aes(x=timeSincePreviousVisit)) +geom_histogram(binwidth = 1)
```



```
#it appears that visit frequency drops after 2 seconds, but it is not a clear drop.

#Most of the literature does a cut off at 2 or 3 seconds.

#Filter dataset to remove visits that were within 2 seconds of eachother

#note that there are NAs because it was the first visit of the dataframe so we can replace that
with 'firstVisit'
df4$timeSincePreviousVisit[is.na(df4$timeSincePreviousVisit)] <- 'firstVisit'

#to ensure we do not loose these visits - i think???? NO THIS JUST DUPLICATES DATA
#df6<-df4%>%
  #filter(timeSincePreviousVisit=='firstVisit')

df6<-df4%>%
  filter(timeSincePreviousVisit>2)
#we need to make the column timeSincePreviousVisit a character back from numerical in order to
bind with df5, which is a character string
df6$timeSincePreviousVisit <- as.character(df6$timeSincePreviousVisit)

df7<-bind_rows(df6)%>%
  arrange(dateTime, TagID_hex,feeder)
view(df7)



write.table(df7, file = "Masterdf_noRepeats.txt",sep="\t",row.names=FALSE)

rm(list = ls(all.names = TRUE))
```

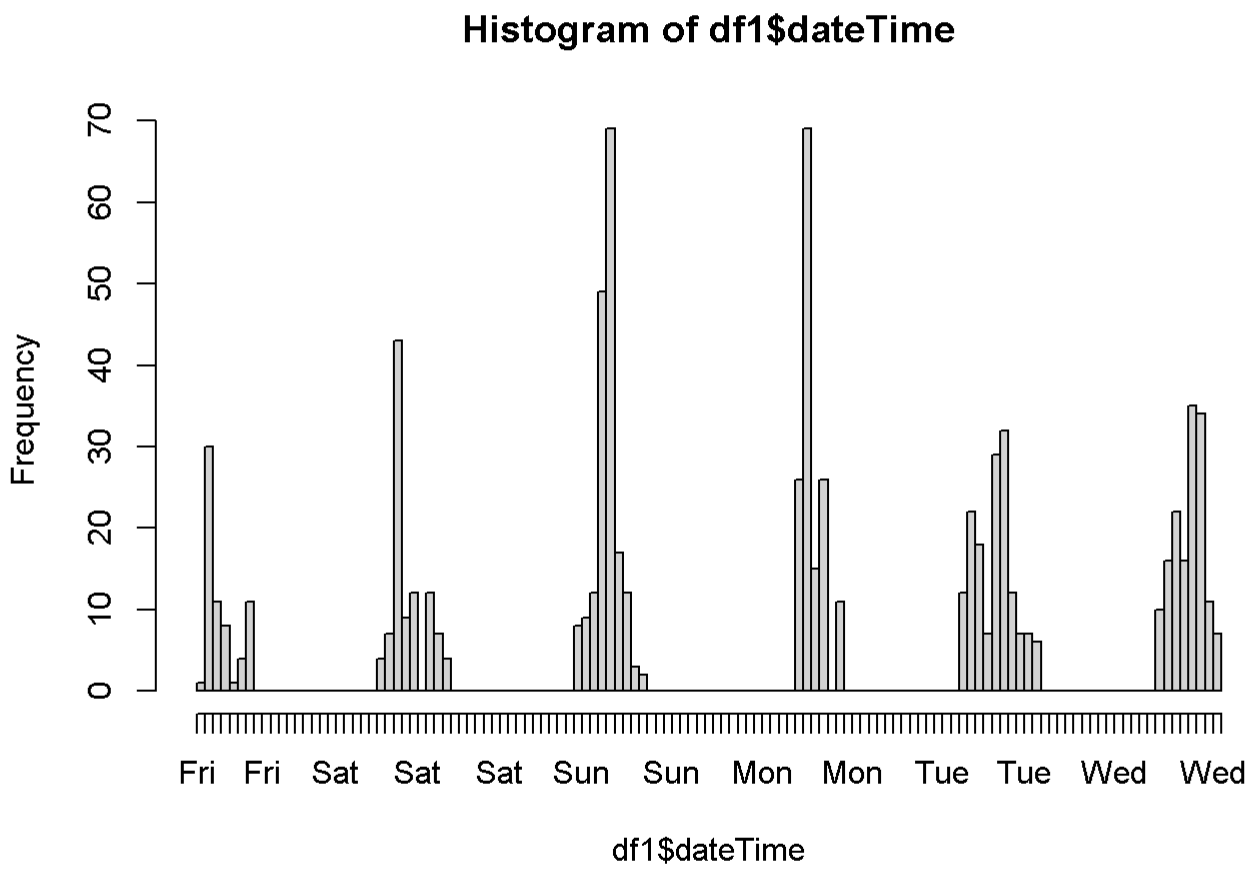
3.3 Check for RFID malfunction. In otherwords, periods of missing data.

we have a dataframe with time since previous visit grouped by individual and by date. We would expect big time gaps across all individuals if a feeder was down.

```
#call the dataframe from your working directory
df1<-read.delim("Masterdf_noRepeats.txt", header=TRUE)

#make dateTime a PosiX class object
df1<-df1%>%
  mutate(dateTime=ymd_hms(dateTime))

class(df1$dateTime)
```

```
## [1] "POSIXct" "POSIXt"
```

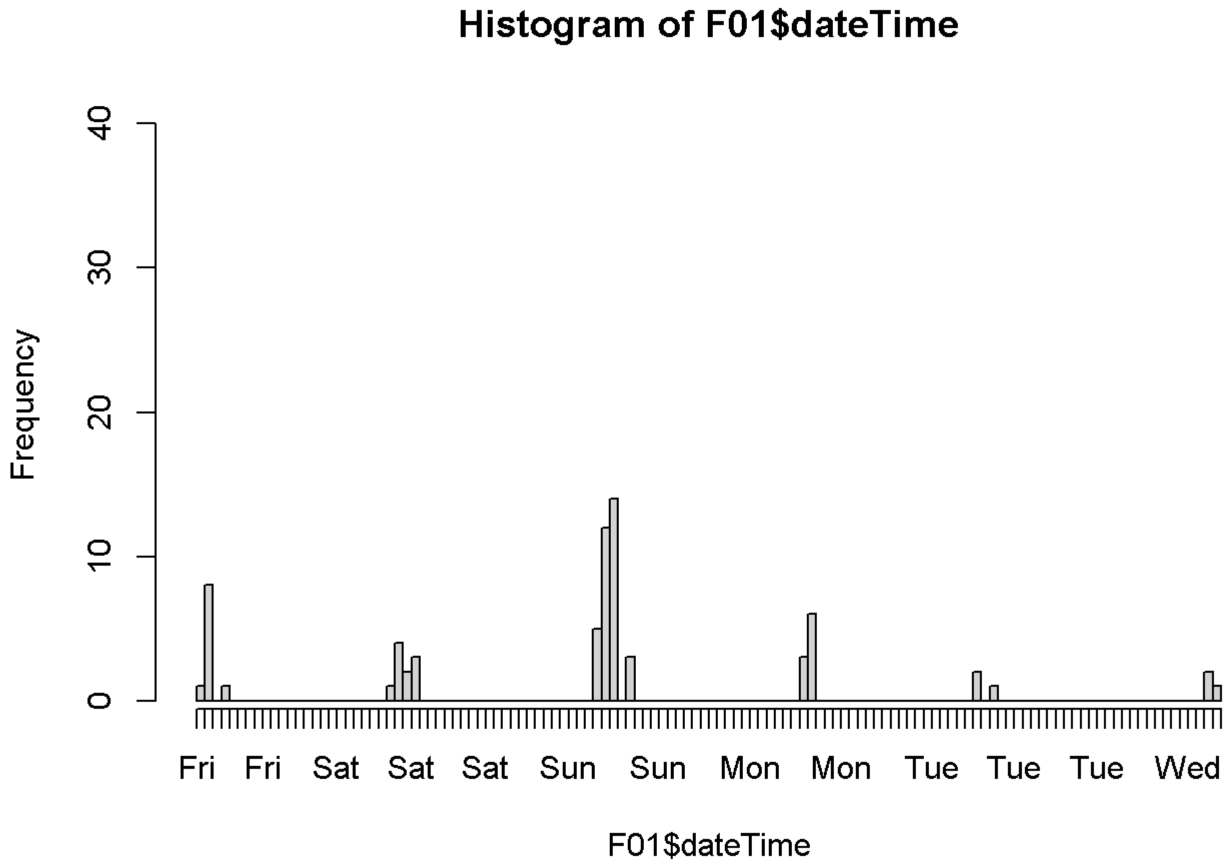
```
#plot a histogram of visit frequency over hours. I think days of week is just assigned to start
on friday.
hist(df1$dateTime, breaks = "hours", plot = TRUE, freq = TRUE, format)
```



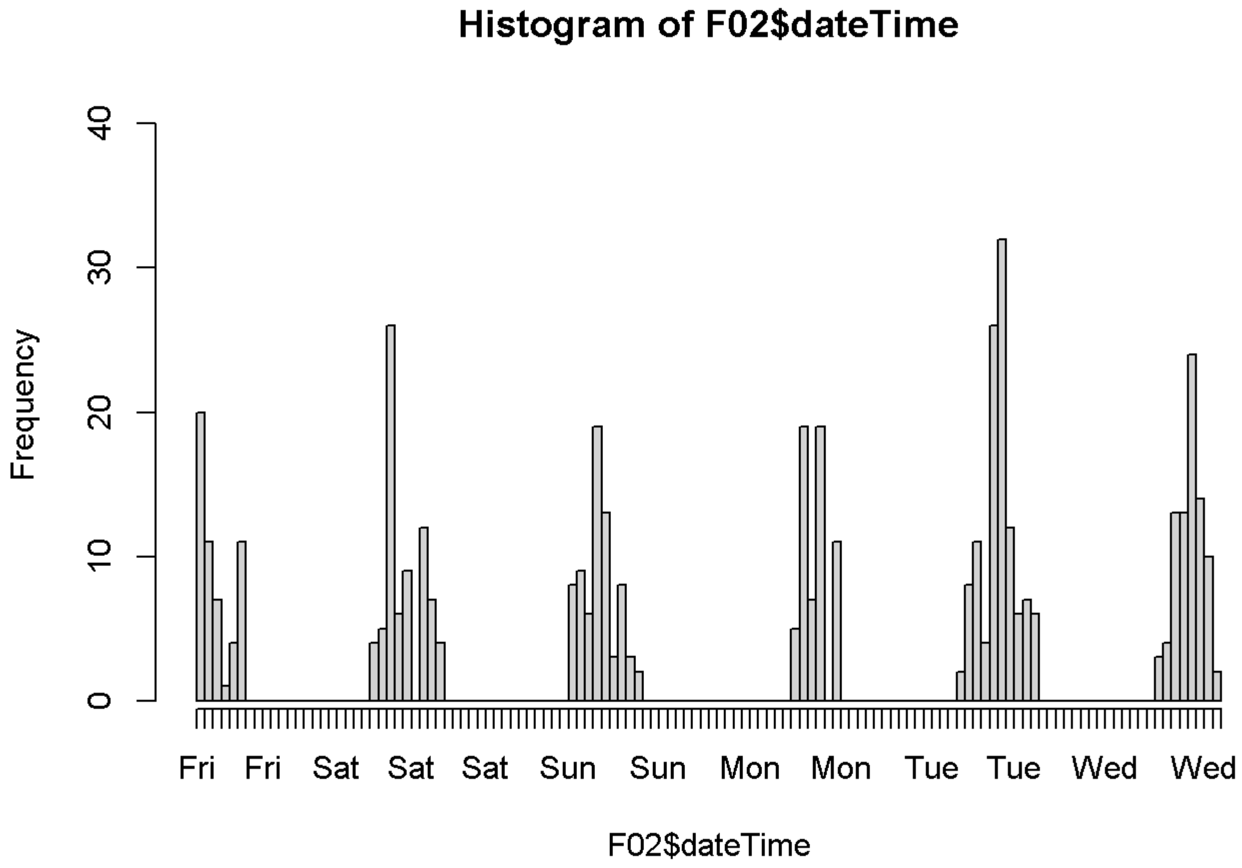
We see there are periods of inactivity but that is regular and likely corresponds to nightime hours.

However, this plot does not indicate whether a given feeder was inactive, as it lumps all three feeders together.

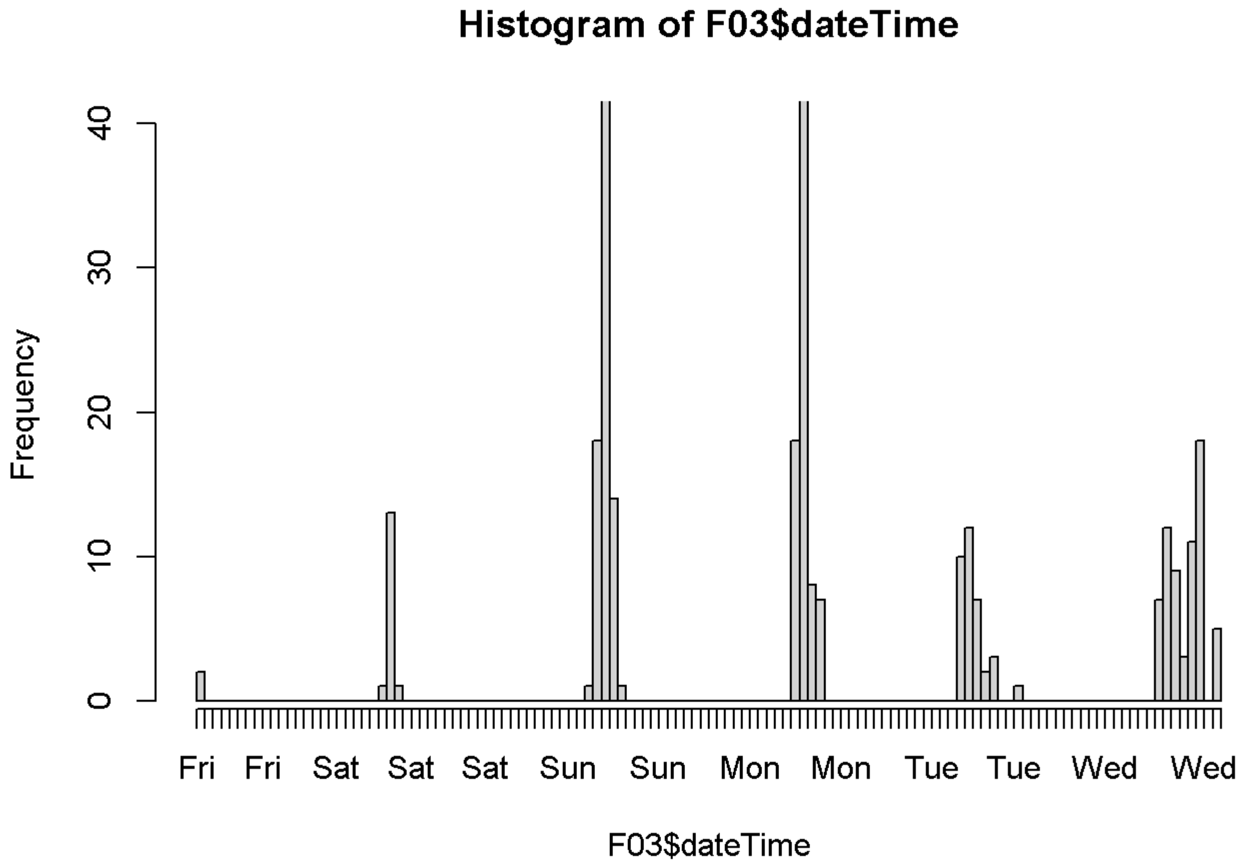
```
#create new dataframes filtered by feeder.

F01<-df1%>%
  filter(feeder=='F01')

#plot a histogram of visit frequency over hours for feeder F01
hist(F01$dateTime, breaks = "hours", plot = TRUE, freq = TRUE, format, ylim=c(1, 40) ) #ylim is
set because I already visualised all the plots and chose to standardise the y axis so they are
comparable. You will have to edit this for your own data.
```



```
F02<-df1%>%
  filter(feeder=='F02')

#plot a histogram of visit frequency over hours for feeder F02
hist(F02$dateTime, breaks = "hours", plot = TRUE, freq = TRUE, format, ylim=c(1, 40))
```



```
F03<-df1%>%
  filter(feeder=='F03')

#plot a histogram of visit frequency over hours for feeder F03
hist(F03$dateTime, breaks = "hours", plot = TRUE, freq = TRUE, format, ylim=c(1, 40))
```



These three feeders have very different distributions of visits, and so this is very suspicious. But there doesn’t seem to be a specific day where feeders are entirely down, except perhaps the first day of the series on F03.

You will need to consider your experimental design and what is expected. In fact this data is from feeders that contained different food types and one feeder was only available for a short time period compared to the others.

3.4 Removing periods of missing data.

If one of the feeders malfunctioned for a day and the entire experiment depends on all feeders working in tandem, this may be good reason to remove data for this day, but see below.

```
#first check the dates data was collected
unique(df1$date)
```

```
## [1] "2022-02-04" "2022-02-05" "2022-02-06" "2022-02-07" "2022-02-08"
## [6] "2022-02-09"
```

```
#assuming feeder03 had a malfunction on the first day (2022-02-04), remove rows that meet the c
onditions of having the following dates in the column "date"
df2<-df1[!(df1$date=="2022-02-04") ,]
#the date should no longer be in the dataframe
unique(df2$date)
```

```
## [1] "2022-02-05" "2022-02-06" "2022-02-07" "2022-02-08" "2022-02-09"
```

```
#if you needed to remove multiple date, the syntax is as follows:

df3<-df1[!(df1$date=="2022-02-04" |df1$date=="2022-02-05" |df1$date=="2022-02-06" ),]
unique(df3$date)
```

```
## [1] "2022-02-07" "2022-02-08" "2022-02-09"
```

How to deal with missing data depends on the experiment and analysis.

Examples:

- For social network analysis, the Machine Learning algorithms workout flocking events from the streams of data over periods of weeks and therefore these analyses are less sensitive to malfunctions, provided you have sufficient periods when the feeders are working
- If you were calculating frequency of nestbox visits per day, and the device failed part of the day, then you would consider not including it that day, or correcting for malfunction time (e.g. frequncy per hour, not by day, or corrected for hours working)
- If you had a learning experiment where birds had access to one out of an array of feeders, for example, Reichert et al 2020 (https://doi.org/10.1098/rsos.192107), they retained data and added a co-variable for feeder malfunction (assigned, not assigned) in their statistical analyses. *We therefore included the duration of feeder malfunction before the bird reached learning criterion for both the assigned (own) feeder and separately for any of the other feeders in that site as additional fixed effects.*

An alternative approach would be to add a column for each feeder and indicate whether it had malfunctioned for each row.

```
#create a new dataframe where it indicates that feeder 03 malfunctioned.
df4<-df1%>%
  mutate(feeder03Malf = case_when(date =='2022-02-04' ~'Y',
                                  date =='2022-02-05'~'Y' ,
                                  date =='2022-02-06'~'Y'))
```

END OF 3. RFID Tutorial: Introduction to RFID data filtering

```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.43      magrittr_2.0.3  stringi_1.7.12  rmarkdown_2.22
##  [5] lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0   dplyr_1.1.2
##  [9] purrr_1.0.1     readr_2.1.4     tidyr_1.3.0     tibble_3.2.1
## [13] ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] highr_0.10       bslib_0.5.0      compiler_4.2.3   pillar_1.9.0
##  [5] jquerylib_0.1.4  tools_4.2.3      digest_0.6.31    timechange_0.2.0
##  [9] jsonlite_1.8.4   evaluate_0.21    lifecycle_1.0.3  gtable_0.3.3
## [13] pkgconfig_2.0.3  rlang_1.1.0      cli_3.6.1        rstudioapi_0.14
## [17] yaml_2.3.7       xfun_0.39        fastmap_1.1.1    withr_2.5.0
## [21] hms_1.1.3        generics_0.1.3   sass_0.4.6       vctrs_0.6.1
## [25] grid_4.2.3       tidyselect_1.2.0 glue_1.6.2       R6_2.5.1
## [29] fansi_1.0.4      farver_2.1.1     tzdb_0.4.0       scales_1.2.1
## [33] htmltools_0.5.5  colorspace_2.1-0 labeling_0.4.2   utf8_1.2.3
## [37] munsell_0.5.0    cachem_1.0.8
```